用于等维分解的 *Direttissimo*算法

Christian Eder * Pierre Lairez † RafaelMohr *‡ Mohab Safey El Di n ‡

2023年6月12日

arXiv:2302.08174v2 [cs.SC] 9 Jun 2023

摘要

我们描述了一种递归算法，可将代数集分解为局部闭等维集，即每个集合的不可约分支都具有相同的维数。在该算法的核心，我们将三角集理论（又称正则链）的思想与格罗布纳基相结合，以编码和处理局部闭代数集。有了这一工具，我们的算法无需对所分解的代数集进行投影，也不需要在分解多项式系统时经常采用的某些一般性假设，例如关于诺特位置的假设。因此，在处理结构性更强的系统时，该算法能够生成细致的分解；而强行确保一般性假设往往会破坏当前系统的结构。实际实验表明，与最先进的实现相比，该算法具有较高的效率。

1 引言

问题陈述 设 $\mathbb{K}$ 是一个代数闭域，令 $R = \mathbb{K}[x_1, \ldots, x_n]$ 为多项式环，并令 $f_1, \ldots, f_c \in R$ 为生成理想 $I \subseteq R$的多项式系统。多项式 $f_1, \ldots, f_c$ 在 $\mathbb{K}$中的零点集 X 唯一地分解为若干不可约代数集的并，且其中任何一个都不包含另一个。这些代数集是 X 的不可约分支，并对应于I的极小关联素理想。若 X 的所有不可约分支具有相同维数，则称代数簇 X 为等维 。显然，X 总能分解为$X = Y_1 \cup \cdots \cup Y_c$，其中$Y_i$ 均为等维代数集。

设 $X = \bigcup_{si=1} X_i$ 是 X 按不可约分支的分解。对 $k = 0, \ldots, n$，定义$Y_k := \bigcup_i \text{s.t. } \dim_{X_i=k} X_i$，则每个 Y{v33>均为等维，因此$X = \bigcup_{nk=0} Y_k$ 是X的等维分解 。给定$f_1, \ldots, f_c \in R$，我们的目标是计算X的这种等维分解。

显然，我们的算法只需在 $\mathbb{K}$ 包含$f_1, ...f_c$ 各系数的子域上进行算术运算，输出也将定义在同一子域上。为使表述简洁，以下我们只在 $\mathbb{K}$ 上工作。

这一问题在机构设计中基于传感器的控制器的奇异性分析（）、实代数几何（）和实代数（）算法，以及自动定理证明和几何学（）中都有自然应用。

先前工作 这一计算问题的重要性催生了大量相关研究，通常也将其作为理想的主分解或代数簇的素分解的中间步骤。针对代数集等维分解的算法，可根据其用于表示（等维）代数集的数据结构进行分类。

使用格罗布纳基进行等维分解有两种常见的主要策略，这些策略经常在计算机代数系统中实现。第一种策略使用代数消元技术。它将输入多项式所生成的理想的维数信息与消元

*德国 RPTU Kaiserslautern-Landau
†Inria、巴黎-萨克雷大学，法国帕莱索
‡索邦大学、法国国家科学研究中心，法国巴黎

定理（Cox 等人，2015年，定理 3.1.2）相结合，以计算所研究代数集在适当仿射线性子空间上的投影描述，从而推断如何分裂相应的理想（）。所研究代数集的最高维等维分支（通常称为 等维包）的投影将由一个超曲面截出，其定义多项式的次数等于该等维包的次数。因此，这类算法的缺点是需要处理次数接近输入方程组贝祖界的多项式。为规避这一缺点，Eisenbud 等人（1992年）提出了另一组称为 直接方法的方法。它们依靠同调代数，将等维分解问题化为协同关系的计算，然后利用这些协同关系分裂所研究的多项式理想，同时避免使用投影。这些算法通常为理想的主分解提供中间步骤。针对这一问题，人们还为 $\mathbb{K}$ 是有限域的情形设计了模技术和专用算法（）。

另一类研究采用代数集的惰性 表示。其核心思想通常是利用这样一个事实：任何等维代数集几乎处处局部都是完全交；也就是说，余维为c 的等维代数集，在其自身的一个稠密扎里斯基开子集上，可表示为c 个多项式的零点集。因此，等维代数集可以理解为由多项式方程和不等式定义的局部闭集的扎里斯基闭包。从这一角度出发，还会要求这c 个多项式构成一个三角集。三角集源自 Wu-Ritt 特征集 ()。其中，三角性是相对于底层多项式环的变量而言的（即在某种意义上类似于线性代数中的三角矩阵）。这种三角结构还自然地给出了代数集的方程，用以描述这c 个多项式无法定义该代数集的情形，因此，三角集以一种自然的方式附带了前文提到的扎里斯基开子集的描述。由于具有三角结构，它们可以将某些算法难题化简为单变量问题。尤其在等维分解领域中，有一种重要的特殊三角集称为正则链，由 **<空引用>** 引入。正则链刻画一个无混合维数的理想，并且对于其所表示的理想具有良好的算法性质。Lazard（1991）也提出了一种相关且常用的算法。使用正则链的算法是计算机代数系统 Maple () 的重要组成部分。关于该主题的介绍，参见 **<空引用>**；关于不同三角集概念之间关系的理论论述，参见 Aubry 等人（1999）。

值得注意的是，与基于格罗布纳基结合代数消元的方法类似，这些三角表示所用多项式的次数在最坏情况下可能高达其所表示的等维分支的次数。尽管如此，与格罗布纳基技术相比，基于三角表示的算法表现往往相当良好，尤其适用于某些稀疏多项式系统。

另一种可自然表示等维代数集的数据结构是几何分解，由 **<空引文>** 提出。几何分解是处于诺特位置的代数集的一种特定零维参数化。在本文的框架中，这些零维参数化用于表示所研究代数集的等维分支中的一般点（这种表示的数值对应物称为见证集（Sommese 等，2005）；本文也将用到这一概念）。在输入满足某些一般情况下成立的假设时，这些参数化可以与直线程序 结合，从而得到目前已知的等维分解的最佳复杂度界。相关方法还可参见（Jeronimo 与 Sabia，2002）。

为绕过“投影次数”问题，人们研究了将增量方法与格罗布纳基算法相结合的方法。这里的“增量”是指：依次向分解算法输入一个输入多项式，这与 Lazard（1991）或 Lecerf（2000）等人的做法相同，用于确定某个多项式何时是由前面多项式生成的理想所商得的多项式环中的零因子。Moroz（2008）将格罗布纳基计算与通过局部闭集表示等维代数集的方法相结合。在此前的工作中，我们也研究了这种方法，利用基于签名的格罗布纳基算法的性质来增强对零因子的检测和利用，并计算多项式系统所谓的非退化轨迹（Eder 等，2022）。

本文工作 在本文中，我们再次采用前文提到的增量方法。与其他增量算法一样，我们的算法以分解算法为基础：给定一个等维代数集 X 和某个 $f \in R$，确定 f 在 X 上的适当交的轨迹，即点 $p \in k^n$ 的集合，其中 $X \cap V(f)$ 的维数比 X 少一维。随后利用该结果遍历输入方程 f_1，。 . . ，f_r。更准确地说，首先对 $V(f_1, f_2)$ 进行分解，然后利用输出结果对 $V(f_1, f_2, f_3)$ 进行分解，依此类推。

与许多其他基于格罗布纳基的等维分解算法不同，我们借鉴三角集理论，像 Moroz（2008）那样使用局部闭集而非多项式理想。在上文概述的迭代方法中，这样做有两项优势。

首先，这种做法能自然地从迭代算法的输出集合中移除分解过程中出现的某些嵌入分支。下面的例子可以说明这一点：

例 1.1。设 $\mathrm{R} := \mathbb{Q}[\mathrm{x}, \mathrm{y}, \mathrm{z}], \mathrm{X} := \mathrm{V}(\mathrm{xy}), \mathrm{f} := \mathrm{xz}$。要将 $\mathrm{X} \cap \mathrm{V}(\mathrm{f})$ 分解为等维分支，可以先将 $\mathrm{X} = \mathrm{V}(\mathrm{x}) \cup \mathrm{V}(\mathrm{y})$ 分解。然后将这两个分支分别与 $\mathrm{V}(\mathrm{f})$ 相交，得到等维分解 $\mathrm{X} \cap \mathrm{V}(\mathrm{f}) = \mathrm{V}(\mathrm{x}) \cup \mathrm{V}(\mathrm{y}, \mathrm{xz})$。后一个集合包含不可约分支 $\mathrm{V}(\mathrm{y}, \mathrm{x})$，它嵌入在 $\mathrm{V}(\mathrm{x})$ 中。若改为将其分裂为不交并 $\mathrm{X} = \mathrm{V}(\mathrm{x}) \cup [\mathrm{V}(\mathrm{y}) \setminus \mathrm{V}(\mathrm{x})]$，并再次将两个分支分别与 $\mathrm{V}(\mathrm{f})$ 相交，则得到 $\mathrm{X} = \mathrm{V}(\mathrm{x}) \cup (\mathrm{V}(\mathrm{y}, \mathrm{z}) \setminus \mathrm{V}(\mathrm{x}))$，且后一个分支不再包含不可约分支 $\mathrm{V}(\mathrm{y}, \mathrm{x})$。

其次，迭代式等维分解算法可能会产生冗余分支；如果不对它们去重，分支数量可能会指数级膨胀：若已将 $X = \bigcup_i X_i$ 分解，而 X_i 之间共享大量不可约分支，那么分别分解每个 $X_i \cap V(f)$ 以获得 $X \cap V(f)$ 的分解，结果会产生更多冗余的分解。由于我们使用局部闭集来表示等维集，因此能够确保每次分解一个局部闭集时，所得输出集两两集合论意义上不交。实验表明，这似乎能在各个分支之间确保足够强的非冗余性，从而避免分支数量指数级膨胀。

本文提供了两种处理算法中出现的局部闭集的方法：一种方法以“朴素”的方式对它们建模，即通过存储其定义方程和不等式来编码，并使用其关联理想的格罗布纳基进行算法处理。另一种方法则尽可能避免必须求出与局部闭集关联的理想的格罗布纳基，转而存储该局部闭集的见证集 的格罗布纳基。在此处使用按分次逆字典序排列的格罗布纳基，与使用三角集的算法相比，使我们能够

- 避免计算待分解代数集的投影，并避免某些常见的一般性假设，例如理想处于诺特位置；
- 以次数较低的多项式描述这些集合。

我们进一步借鉴三角集理论，并采用这样一种启发式原则：处理给定的代数集时，尽可能频繁、尽可能精细地对其进行分解。这一理念融入了我们算法的递归结构之中；该结构旨在给定某些饱和理想的生成集后，尽可能充分地分解给定的局部闭集。

我们在计算机代数系统 Oscar（OSCAR 团队，2023）中实现了我们的算法，并通过其接口调用 msolve 库（Berthomieu 等人，2021）完成所有必要的 Gröbner 基计算。实验结果表明，我们的算法能够处理一些多项式系统，而这些系统超出了主流计算机代数系统中现有等维分解算法实现的能力范围。

2 算法

2.1 原理

为了说明我们等维分解算法背后的基本原理，考虑仿射空间 $\mathbb{K}^n$中的等维代数簇X 。设$f \in R$。代数簇X 划分为：

1. 在p 处局部而言p 满足f 是非零因子的点（即在环$R_p/I(X)R_p$中）。多项式f 在p 于X中的任意开邻域内都取非零值。这定义了X^{proper}，它是X的一个开子集。2. 包含在X 的某个不可约分支中的点p，在该分支上f 恒等于零。这定义了X^{improper}，它是X的一个闭子集。

显然，$X = X_{\text{proper}} \sqcup X^{\text{improper}}$（其中 $\sqcup$ 表示不交并），并且 $X_{\text{improper}} \subseteq V(f)$，因此

$$X \cap V(f) = \big(X_{\text{proper}} \cap V(f)\big) \sqcup X_{\text{improper}}.$$

根据构造，$X_{\text{proper}} \cap V(f)$ 是一个正常 交：它是等维的，维数为 $\dim X - 1$，或为空。作为X的不可约分支之并，闭集 X^{improper} 是等维的，其维数与X相同，除非它为空。因此，我们得到$X \cap V(f)$的一个等维分解。给定X的定义方程后，可以迭代应用此过程，得到任意仿射代数簇的等维分解。

在我们的算法中，我们应用上述想法，而不直接计算 $X^{textproper}$ 和 X^{improper}。令 $I(X) \subset R$ 为一个理想，使得 $V(I(X)) = X$。此外，我们用 $(I(X) : f^\infty)$ 表示 $I(X)$ 关于 f 的饱和理想。回想一下，$V((I(X) : f^\infty))$ 是 $X \setminus V(f)$ 的扎里斯基闭包（Cox 等人，2015年，定理 4.4.10）。

我们寻找一个元素 $g \in (I(X) : f^\infty) \setminus I(X)$。如果不存在这样的元素，这意味着 $X_{\text{improper}} = \varnothing$，因此 $X \cap V(f)$ 是等维的。如果存在这样的 g，则考虑 X 的如下划分：

1. 闭轨迹 X_1，由满足以下条件的点 p 组成：在 X 中，g 在 p 的任意邻域内都取非零值；

2. 开轨迹 X_2，由满足以下条件的点 p 组成：在 X 中，g 在某个 p 的邻域内恒为零。

这两个集合都是等维的。根据构造，fg 在 X 上恒等于零，因此 $X_1 \subseteq X^{\text{improper}}$，这就给出了 X 的如下分解：

$$X = X_1 \sqcup (X_2 \cap V(f)). \tag{1}$$

X_1 的理想为 $(I(X) : g^\infty)$。项 $X_2 \cap V(f)$ 可能不是等维的，但我们可以递归地应用上述思路：如果存在一个元素，我们就再次沿着 $(I(X_2) : f^\infty) \setminus I(X_2)$ 中的一个元素将 X_2 分裂。这将导出算法 *split*。

集合 X_2 不是闭集，因此我们不仅需要处理仿射空间中的闭集，还需要更一般地处理局部闭集。为此，我们将它们划分为特殊的局部闭集，更确切地说，划分为仿射空间中某个超曲面的补集内的闭集，我们称之为仿射胞腔。具体而言，设$I(X_1) = (I(X) : g^\infty)$ 由有限生成集$H \sqcup \{h\} \subset R$给出。接着，我们通过$X_2 = X \setminus V(H \sqcup \{h\})$ 递归地分解

$$X_2 = X \setminus V(H \cup \{h\}) = (X \setminus V(h)) \sqcup \big((X \setminus V(H)) \cap V(h)\big).$$

与$V(h)$ 的交集使用*split* 计算，以确保等维性。下面的算法*remove*执行这些操作。最后，我们通过重复应用 *split*，按照增量策略得到一个等维分解算法，见算法*equidim*。

接下来介绍我们用于操作仿射胞腔的基本操作，而算法的正确性证明和终止性见第 2.3 节。

2.2 基本操作

定义 2.1. 仿射胞腔 X 是 $\mathbb{K}^n$ 中形如 $Z \setminus V(g)$ 的局部闭集，其中 Z 是代数集，且 $g \in R$。若扎里斯基闭包 X 的所有不可约分支具有相同的维数，则称仿射胞腔 X 是等维的。

无论仿射胞腔的表示方式如何，我们都假设可以对任意仿射胞腔X执行以下操作：

(1) 给定$f \in R$，计算仿射胞腔 $X \cap V(f)$；

(2) 给定$f \in R$，计算仿射胞腔$X \setminus V(f)$；

与有效代数几何中的常见情况一样，代数集由理想定义，而这些理想并不总是根基，因此我们的仿射胞腔都配有一个指定理想 I(X) $\subseteq$ R，使得 X $=$ V(I(X))。I(X) 的根基记作 rad I(X)。我们假设操作 (1) 和 (2) 满足 I(X) $+\langle f\rangle \subseteq$ I(X $\cap$ V(f)) 和 I(X) $\subseteq$ I(X $\setminus$ V(f))。此外，我们假设可以对任意仿射胞腔 X 执行以下操作：

(3) Given $f \in R$, decide if $f \in I(X)$;

(4) 计算$I(X)$的一个基，记为$basis(X)$。

例如，我们可以用一对 (F, g) 表示仿射胞腔 X，其中 F 是 I(X) 在某个单项式序下的一个格罗布纳基，g 是一个满足 X $=$ X $\setminus$V(g) 的多项式（关于格罗布纳基的介绍，参见 Becker & Weispfenning，1993）。我们将 X $=$ V(F; g) 记作。对于集合 F $\subseteq$ R 和元素 g $\in$ R，令 sat(F, g) 表示饱和理想 ($\langle$F$\rangle$: g$^\infty$) 的一个格罗布纳基。回忆

$$(I : g^\infty) \stackrel{\text{def}}{=} \left\{ f \in R \;\middle|\; \exists k \in \mathbb{N}, f g^k \in I \right\}.$$

利用这些基本的sat，我们可以执行上述所有四种运算：

(1) V(F; g) $\cap$V(f) $=$V(sat(F $\cup\{$ f$\}$, g);
g); (2) V(F; g) $\setminus$ V(f) $=$V(sat(F, f); f g);

(3) $f \in I(X)$ 当且仅当f 关于F 的标准形为零时；

(4) 基$(V(F;g)) = F$。

备注 2.1。 在第 3 节中，我们将说明如何运用一种称为见证集的概念，对仿射胞腔X执行上述基本操作。Lecerf 于 2003 年首次提出这一概念，用于等维分解，当时称之为提升纤维。由此可以更惰性地表示X，这种表示方式并不总是需要$I(X)$ 的格罗布纳基。

算法1 等维分解

输入 一个等维仿射胞腔X，一个元素$f \in R$

输出 将$X \cap V(f)$划分为等维仿射胞腔

```
1  函数分裂(X, f)
2      G ← 基(X \ V(f))
3      如果G ⊆ I(X)        [可替换为 G ⊆ 根基I(X)]
4          返回 {X ∩ V(f)}
5      else
6          g ← G \ I(X)中的任意元素
7          H ← 基(X \ V(g))
8          D ← {X ∩ V(H)}
9          对 Y ∈ 移除(X ∩ V(g), H)
10             D ← D ∪ 分裂(Y, f)
11         end
12         返回 D
13     end
14 结束
```

输入 一个仿射胞腔X，一个有限集$H \subset R$

前置条件 $X \setminus V(H)$ 是等维的

输出 将$X \setminus V(H)$ 划分为等维仿射胞腔

1 function remove(X, H) 2 if H = ∅3 return ∅4 else 5 h ← H 的任意元素6 **D ← {X \V(h)} 7 for Y ∈ remove(X, H \ {h}) 8 D ← D ∪ split(Y,h) 9 end 10 return D11 end 12 end**

输入 一个有限集 $F \subseteq R$

输出 $V(F)$ 划分为等维仿射胞腔

1 **function***equidim*(F)2 $\mathcal{D} \leftarrow \{V(\varnothing;$ 1)} [全仿射空间]3 **for** f **in** F 4 $\mathcal{D} \leftarrow \bigcup_{X \in \mathcal{D}} split(X, f)$5 **end**6 **return** $\mathcal{D}$7 **end**

例 2.1. 为了说明算法 分裂，我们详细展示它对输入 $X := V(xy,zw)$ 和$f := xz$的处理过程。使用算法 分裂 的记号，我们得到 $G = \{y,w\}$。这不包含于 $I(X)$中，因此我们可以在算法 分裂第 6 行选择 $g := y$。接着得到 $H = \{x,zw\}$。注意，$X \setminus V(zw) = \varnothing$，并且

因此算法 分裂返回

$$\begin{aligned} & X \cap V(H) \text{ and } split(remove(X,H),f) \\ = & V(zw,x) \text{ and } split(V(y,zw) \setminus V(x), xz). \end{aligned}$$

对算法 分裂 的第二次调用找到了 $G = \{y、w\}$；同样，这个集合不包含于 $I(V(y、zw) \setminus V(x))$ 中，因此我们可以在第 6 行选择 $g := w$。然后我们找到 $H = \{z\}$，这次得到

$$\begin{aligned} split(V(y,zw) \setminus V(x), xz) = & V(y,z) \setminus V(x) \\ & \text{and } split(V(y,w) \setminus V(xz), xz) \end{aligned}$$

最后一次调用分裂算法只是找到了空集，因此总的来说，我们得到了分解

$$V(xy,zw,xz) = V(x,zw) \cup V(y,z) \setminus V(x).$$

备注 2.2。例 2.1 说明，即使代数集是等维的，算法 分裂 仍可能将其分裂。从启发式角度看，算法 等维 中的中间分解越精细，后续步骤的计算就越容易。

2.3 正确性和终止性

在计算等维仿射胞腔 X 与超曲面 $V(f)$ 的交集时，我们根据 $V(f)$ 是否与 X 正确相交而区分两种情况。引理 2.2 处理第一种情况，而引理 2.3 处理第二种情况。

引理 2.2. 设 X 是一个等维仿射胞腔。设 $f \in R$，使得 $(I(X) : f^\infty) \subseteq$ 根基 $I(X)$。则 $X \cap V(f)$为空，或为等维的，且维数为 维数$X - 1$。

证明。设 I = I(X)。不妨设 I 是根基理想：假设 (I : f$^\infty$) $\subseteq$ 根基 I 也意味着 (根基(I) : f $^\infty$) $\subseteq$ 根基 I。若 a $\in$(根基(I) : f $^\infty$)，则对某个 r ≥ 0，有 a f^r $\in$根基 I，因此对某个 s ≥ 0，有 (a f r)^s $\in$I。特别地，a^s $\in$(I : f$^\infty$) $\subseteq$根基 I。因此 a $\in$根基 I。假设 X $\cap$V(f) 非空。根据 Krull 主理想定理，I $+\langle$ f$\rangle$ 上的任一极小素理想的余维至多为 codim I $+1$。条件 (I : f $^\infty$) $\subseteq$根基 I 在几何~~上意味着~~ X$\subseteq$X $\setminus$ V(f)，因此 f 在 X 中任意点的邻域内都取非零值。所以 f 不是 R/I 中的零因子。特别地，在 I $+\langle$ f$\rangle$ 中存在一个长度为 codim I $+1$ 的正则序列。由于多项式环 R 是 Cohen-Macaulay，因此 I $+\langle$ f$\rangle$ 上的每个极小素理想的余维至少为 codim I $+1$。

□

引理 2.3. 设 X 是一个等维仿射胞腔。设 $f \in R$，设 $g \in (I(X) : f^\infty)$，并设 $I_g = (I(X) : g^\infty)$。令 $X_1 = X \cap V(I_g)$ 且 $X_2 = (X \cap V(g)) \setminus V(I_g)$。则：

(i) $X = X_1 \sqcup X_2$；*(ii)* $X \cap V(f) = X_1 \sqcup (X_2 \cap V(f))$；

(iii) X_1 为空，或为等维，且 维数X_1 = 维数 X；

(iv) X_2 为空，或为等维，且 维数X_2 = 维数 X；

证明。显然 $X = X_1 \sqcup (X \setminus V(I_g))$。作为点集，$X_1$ 是 X 的那些不可约分支的并，在这些分支上 g 不恒为零。特别地，$X \setminus V(I_g)$ 是 X 中这样一些点的集合：在这些点的某个邻域内，g 恒为零。因此 $X \setminus V(I_g) \subseteq V(g)$，于是我们得到

$$X \setminus V(I_g) = (X \cap V(g)) \setminus V(I_g),$$

由此得到 (i)。

接下来，我们有 I(X$_1$) = I(X) + I$_g$ = (I(X) : g$^\infty$)。此外，f $\in$rad I(X$_1$)。事实上，根据 g 的定义，对于某个 k ≥ 0，有 g f^k $\in$I(X)，因此 f $\in$rad (I(X) : g) $\subseteq$rad(I(X) : g$^\infty$)。所以 X$_1$ $\subseteq$V(f)。由此可得 X $\cap$V(f) = X$_1$ $\sqcup$ (X$_2$ $\cap$V(f))。这就证明了 (ii)。

由于X 是等维的，因此X_1 （作为X的不可约分支之并）也具有相同的维数，除非它为空。这证明了 (iii)。至于X_2, ，它是X中的开集，因此继承了X的等维性和维数，除非它为空。这证明了 (iv)。

□

现在我们通过相互归纳证明*split* 和*remove*算法的正确性和终止性。在第 3 行，测试$G \subseteq I(X)$ 可以替换为$G \subseteq \mathrm{rad}I(X)$，或者替换为任何在$G \subseteq I(X)$成立、而在$G \not\subseteq \mathrm{rad}I(X)$不成立的条件；这不会影响正确性或终止性。我们将在第 3 节使用这一变体。

定理 2.4. 对于任意仿射胞腔 X:

(i) 若 X 是等维的，则对于任意 $f \in R$，过程 分裂 在输入 X 和 f 时会终止，并将 $X \cap V(f)$ 划分为等维仿射胞腔 Y，且 $I(X) \subseteq I(Y)$。

(ii) 对于任意有限集 $H \subset R$，若 $X \setminus V(H)$ 是等维的，则过程 移除 在输入 X 和 H 时会终止，并将 $X \cap V(H)$ 划分为等维仿射胞腔 Y，且 $I(X) \subseteq I(Y)$；

证明。 我们对$I(X)$ 应用诺特归纳，并假设对于任意满足X'且$I(X) \subsetneq I(X')$的仿射胞腔，结论均成立。

我们先处理分裂。令$f \in R$，并令$I_f = (I(X) : f^\infty)$。若$I_f \subseteq I(X)$，则引理 2.2 适用，且$X \cap V(f)$ 是等维的。因此，在这种情况下，分裂(X, f) 会终止且正确。

现在假设存在某个 $g \in I_f \setminus I(X)$。令 $I_g = (I : g^\infty)$。引理 2.3 适用：$X \cap V(f)$ 的一个等维分解由 $X \cap V(I_g)$ 以及 $\big((X \cap V(g)) \setminus V(I_g)\big) \cap V(f)$ 的一个等维分解给出。此外，$(X \cap V(g)) \setminus V(I_g)$ 是等维的。由于 $g \notin I(X)$，我们有 $I(X) \subsetneq I(X \cap V(g))$，因此 remove$(X \cap V(g), H)$（使用 split 算法的记号，其中 H 是 I_g 的一个生成集）是正确的，并且根据归纳假设会终止。此外，它会输出仿射胞腔 Y，使得 $I(X) \subsetneq I(X \cap V(g)) \subseteq I(Y)$。因此，递归调用 split$(Y, f)$ 是正确的，并且会终止。

至于移除，令$H \subset R$ 为有限集，使得$X \setminus V(H)$ 是等维的。若$H = \varnothing$，则 (ii) 显然成立。对于$H \neq \varnothing$的情形，令 $h \in H$ 且$H' = H \setminus h$。由于$V(H) = V(h) \cap V(H')$，因此我们有

$$X \setminus V(H) = (X \setminus V(h)) \sqcup \big((X \setminus V(H')) \cap V(h)\big). \tag{2}$$

集合 $X \setminus V(h)$ 和 $X \setminus V(H')$ 在 $X \setminus V(H)$ 中是开集，因此是等维的（或为空）。对 H 的基数进行归纳，假设 remove(X, H') 将 $X \setminus V(H')$ 划分为等维仿射胞腔，并且该划分中的每个胞腔 Y 都满足 $I(X) \subseteq I(Y)$。根据 (i)，调用 split(Y, h) 会终止，并将 $(X \setminus V(H')) \cap V(h)$ 划分为满足 $I(X) \subseteq I(Y)$ 的胞腔 Y。此外，仿射胞腔 $Y = X \setminus V(h)$ 也满足 $I(X) \subseteq I(Y)$。根据 (2)，remove(X, H) 也会终止，并将 $X \setminus V(H)$ 划分为满足 $I(X) \subset I(Y)$ 的胞腔 Y。 □

推论 2.5. 算法等维 正确且会终止。

3 实现与实验结果

3.1 实现细节

本节介绍一些实现细节和替代方案。具体而言，我们展示一种更惰性的仿射胞腔数据结构，它能够以蒙特卡洛随机化为代价，延迟部分格罗布纳基计算。我们既实现了第 2 节中描述的方法，也实现了本节中描述的方法。

对于任一种方法，我们都需要一种算法：给定理想 I 的生成元和一个元素$f \in R$，计算饱和$(I : p^\infty)$的生成元。即使采用惰性表示，有时仍需要这样做，以便计算理想$I(X)$的格罗布纳基，其中X 是一个仿射胞腔。在概率设置下，某些饱和将由零维理想的饱和取代。

在我们的实现中，我们选择了使用格罗布纳基进行饱和的标准方法。为了计算 $(I : p^\infty)$ 的生成元，针对新变量 t ，在 $R[t]$ 上固定一个单项式序 $\leq$ ，使得 $\leq$ 消去 t。针对 $\leq$，计算理想 $I + \langle tp - 1\rangle \subset R[t]$ 的格罗布纳基 G。根据消去定理，G 中不含变量 t 的元素构成 $(I : p^\infty)$ 的格罗布纳基。其他饱和方法也存在，例如 Eder 等人（2022）或 Berthomieu 等人（2022）提出的方法。

随机化依赖于与适当维数的随机线性子空间求交，从而将问题化归到零维情形。这一思想在符号计算（Lecerf，2003）和数值代数几何（例如 Bates 等人，2013）中广为人知；在这些领域中，代数集与随机选取的适当随机线性子空间的交被称为见证集。

命题 3.1. 设 $X \subseteq \mathbb{K}^n$ 是一个维数为 d 的等维仿射胞腔，且 $f \in R$。那么，对于余维为 d 的一般线性子空间 $L \subset \mathbb{K}^n$，以下陈述成立：

1. $f \in \mathrm{rad} I(X)$ 当且仅当 $f \in \mathrm{rad}\, I(X \cap L)$。

2. $I(X \setminus V(f)) \subseteq \mathrm{rad}\, I(X)$ 当且仅当 $X \cap L \cap V(f) = \varnothing$。

证明。我们总有 rad I(X) $\subseteq$ rad I(X $\cap$ L)。反过来，设 f $\notin$ rad I(X)。令 U $=\{$p $\in$ X $\mid$ f (p) $\neq 0\}$/。根据假设，它是 X 的一个非空开子集。由于 X 是等维的，U 的维数为 d，且交集 U $\cap$L 非空（因为 L 是一般的）。因此，f 在 X $\cap$ L 的一个非空子集上不为零。特别地，f $\notin$ rad I(X $\cap$ L)。这就证明了第一点。

关于第二点，注意到 I(X $\setminus$ V(f)) $\subseteq$ rad I(X) 当且仅当 X 与 V(f) 正确相交，也就是说，X $\cap$ V(f) 是维数为 d $-$ 1的等维对象。X $\cap$ V(f) 与余维为 d 的一般空间 L 的交为空，当且仅当 X $\cap$ V(f) 的维数小于 d。这就证明了第二点。 □

在此设定下，我们用四元组 (F, G, W, d) 表示一个等维仿射胞腔 X，其中 F、G 和 W 是 R 的子集，d 是整数，并且 dim X $=$ d，X $=$ V(F) $\setminus$ V($\prod_{g\in G}$ G)，而 W（表示见证集）是某个余维为 d 的一般线性子空间 L 的 I(X $\cap$ L) 的格罗布纳基。我们将 X $=$ V(F; G, W, d) 记作如此。实际上，L 只是随机选取的，只有在高概率下才能满足充分的一般性（假设 $\mathbb{K}$ 有足够多的元素）。仅给定 F、G 和 d 时，我们可以通过选取一个由 d 个随机线性形式组成的集合 J $\subseteq$ R，并计算 $(((F : g_1^\infty) : \cdots) : g_r^\infty)$ 的格罗布纳基，从而求得合适的集合 W，其中 G $= \{g_1, \ldots, g_r\}$。此过程记为 witness(F, G, d)。

这四种基本操作的执行方式如下。对于交运算，我们需要了解输出的预期维数。设 $X = V(F; G, W, d)$ 是一个等维胞腔。

(1) *[正常交]* 给定 $f \in R$，使得 X 与 $V(f)$ 正常相交，

$$X \cap V(f) = V\big(F'; G, witness(F', G, d-1), d-1\big),$$

其中 $F' = F \cup \{f\}$；

(1') *[纯非正常交]* 给定 $H \subset R$，使得 $X \cap V(H)$ 是 X 的不可约分支之并，

$$X \cap V(H) = V\big(F \cup H; G, gb(W \cup H), d\big),$$

其中 $gb(W \cup H)$ 表示由 $X \cup H$生成的理想的格罗布纳基；

(2) for $f \in R$, $X \setminus V(f) = V(F; G \cup \{f\}, sat(H, f), d)$;

(3) $f \in$ 当且仅当 $1 \in (W : f^\infty)$ 时，$I(X)$ 为根理想。

(4) $I(X)$ 通过依次用 G 的所有元素对 $\langle F \rangle$ 进行饱和运算来计算。

在分解算法中，我们总是预先知道每个交的类型。split 第 4 行的交是正常交，第 8 行的交是纯非正常交。第 9 行的交则更微妙。事实上，在考虑 X $\cap$V(g) 时，分解算法可能会在这里产生一个非等维胞腔。用该算法的记号来说，胞腔 X′ $=$ X $\cap$V(g) 仅在 V(H) 之外是等维的（维数为 dim X）。这个非等维胞腔只会执行四种基本操作中的一种：对于某个 h $\in$ H，执行 X′ $\setminus$V(h)。此操作会恢复等维性。因此，我们基本上可以忽略这个问题，并将交 X $\cap$V(g) 作为纯非正常交来计算，假定 X $\cap$V(g) 是等维的。

算法2 真交检验

输入 一个等维仿射胞腔 X、一个元素 $f \in R$

输出 true 若 $X \cap V(f)$ 是正常交，则输出**false** 否则

```
1 函数 isProper(X, f)
2       W ← 是X的含性集合
3       W′ ← 是(⟨W⟩ : f^∞)的格罗布纳基
4       返回 1 ∈ W′
```

此外，我们还得到第五种操作：一种概率算法，用于检查 $X \cap V(f)$ 是否为空，或是否为维数比 X 小一的等维集（或者等价地，$(I(X) : f^{\infty}) \subseteq \operatorname{rad} I(X)$）。这由算法 *isProper* 给出。借助此算法，我们可以将算法*split* 第 3 行中的 if 条件替换为*isProper*(X, f)。只有在条件不成立时，我们才继续计算$I(X \setminus V(f))$的格罗布纳基。

最后，我们想指出以下几点：在 split 算法中，给定输入 X 和 f，我们可能需要计算 G：＝基(X \ V(f))，但在 split 算法第 6 行中，我们只使用 G 中的一个元素。通过一种简单的缓存机制可以改善这种情况：注意，在 split 算法第 10 行中，我们调用 split(Y, f)，其中 Y 是满足 Y ⊂ X 的仿射胞腔。这当然意味着 G ＝基(X \ V(f)) ⊆基(Y \ V(f))。因此，在计算基(Y \ V(f)) 之前，我们可以先尝试从调用 split(Y, f) 时第 6 行已经计算出的集合 G 中选取一个元素。

3.1.1 新数据结构的设计依据

始终知晓　　仿射胞腔的格罗布纳基，给等维算法带来了很大的成本负担
对.....的成本　我们的算法。

这实际上与关于某些正则性假设下格罗布纳基复杂度的著名观察有关。事实上，对于处于强诺特位置的正则序列，以增量方式计算中间格罗布纳基所需的线性代数步骤，其成本高于最后几步（Bardet 等，2015）。维数相关的复杂度界进一步印证了这一现象（Hashemi 与 Seiler，2017）。

使用见证集，我们有望在算法中避免大量中间格罗布纳基计算。根据我们的经验，在许多情况下，使用见证集能显著提高算法效率，而前文提到的复杂度结果也为此提供了理论依据。

此外，在上一小节介绍的仿射胞腔数据结构中，我们将仿射胞腔的定义不等式存储为因式分解。若要用一个已知具有因式分解的元素 I 对多项式理想 $f \in R$ 进行饱和，其中该因式分解为 $f = \prod_{g \in G} g$，由有限集 G 给出，那么预期逐一用元素 $g \in G$ 进行饱和，会比使用上述消元方法直接用 f 进行饱和成本更低。这会降低所涉及多项式的次数。

3.1.2 *remove* 的改进版本

此外，我们在按照第 2 节所述实现 remove 算法时遇到了以下问题。在该算法中，H 是一个格罗布纳基，因此往往冗余度很高（也就是说，远非极小生成集）。因此，常常会出现两个或更多元素 $h_1, h_2 \in H$，使得对于 $X_1 := X \setminus V(h_1)$ 和 $X_2 := X \setminus V(h_2)$，有 $I(X_1) = I(X_2)$。最终，集合 X_1 和 X_2 会变得不相交，因为最终 X_1 会与 $V(h_1)$ 相交，或 X_2 会与 $V(h_1)$ 相交，但在此之前，split 算法可能不得不将 X_1 和 X_2 分裂。由于使用 split 算法对仿射胞腔进行分裂只取决于底层理想，我们可能会在理想层面重复两次或多次完全相同的操作。由于算法具有递归性质，这个问题会呈指数级加剧。因此，我们修改了 remove 算法，以尽可能快地从 X_1 和 X_2 得到互不相交的等维仿射胞腔，由此得到算法 3。请注意，使用见证集时，该算法无需预先知道理想 $I(X_i)$ 的格罗布纳基，直到可能执行到第 12 行才需要。

算法 3 remove′

输入 一个仿射胞腔 X，一个有限集 $H \subset R$

输出 $X \setminus V(H)$ 的一个划分，由等维仿射胞腔组成

函数 *remove′*(X,H)
  $\mathcal{D} \leftarrow \emptyset$
  对于 i **从** 1 **到**r
    $X_i \leftarrow X \setminus V(h_i)$
    $H_i \leftarrow \emptyset$
    对于 j **从** 1 **到** $i-1$
      如果 *isProper*(X_i, h^j)
        $X_i \leftarrow X_i \cap V(h^j)$
      else
        $H_i \leftarrow H_i \cup \{h_j\}$
      end
    end
    $\mathcal{D}_i \leftarrow X_i \cap V(H_i)$ 的分解
      通过反复应用分裂
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
  end
  返回 $\mathcal{D}$
结束

3.2 实验结果

本节给出一些实验结果。我们在下表中比较了我们对算法 等维 的实现与各种计算机代数系统中用于代数集等维分解的方法的运行时间。下一节将更详细地讨论其中一些运行时间。我们与以下实现进行了比较:

1. Maple 中 RegularChains 库的 Triangularize 函数（Lemaire 等人，2005），该函数将多项式系统分解为正则链；2. Oscar 中的 equidim_decomposition_weak 函数（OSCAR 团队，2023），该函数是相应 Singular 函数的封装；3. Magma（Bosma 等人，1997）中的 EquidimensionalDecomposition 函数（对应表中的“Magma”列）和 ProbablePrimeDecomposition 函数（对应表中的“Magma（素数分解）”列）；以及 4. 数值多项式系统求解器 Bertini（Bates 等人，2013），我们通过请求以固定精度分解为不可约分支的见证集，并将精度设置为 Bertini 的默认值，在每个相关系统上运行了该求解器。

我们算法的实现本身也是在 Oscar 中完成的。Oscar 使用 Julia 编程语言编写（Bezanson 等人，2017）。其源代码可在以下位置获取

https://github.com/RafaelDavidMohr/Decomp.jl

对于所有必要的格罗布纳基计算，我们都使用 msolve 库（Berthomieu 等人，2021），Oscar 为其提供了接口。我们的示例系统包括:

1. Cyclic（8），源自经典的 Cyclic（n）基准测试。

2. 方程组 P4L 1至3 来自机器人学中的四线透视问题，参见 García Fontán 等人（2022）。

3. 方程组 C1 至 C3 是某些单个多元多项式的雅可比理想，这些多项式定义了奇异超曲面。

4. Ps(n)，通过多项式编码伪奇异点

$$f_1, \ldots, f_{n-1}, g_1, \ldots, g_{n-1}$$

其中，$\mathrm{f}_i \in \mathbb{K}[\mathrm{x}_1, \ldots, \mathrm{x}_{n-2}, \mathrm{z}_1, \mathrm{z}_2], \mathrm{g}_i \in \mathbb{K}[\mathrm{y}_1, \ldots, \mathrm{y}_{n-2}, \mathrm{z}_1, \mathrm{z}_2]$，中的 f_i 取为随机稠密二次型，而 g_i 则选取为使 $\mathrm{g}_i(\mathrm{x}_1, \ldots, \mathrm{x}_{n-2}, \mathrm{z}_1, \mathrm{z}_2) = \mathrm{f}$，即取 f_i 在变量 $\mathrm{y}_1, \ldots, \mathrm{y}_{n-2}, \mathrm{z}_1, \mathrm{z}_2$中的副本。

5. sos(s,n)，编码将投影限制在由 $\mathbb{K}[x_1, \ldots, x_n]$ 中 s 个随机稠密二次型之和构成的超曲面上、并投影到第一个坐标时的临界点。

$$f, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}, \quad f = \sum_{i=1}^{s} g_i^2.$$

6. sing(n)，编码将投影限制在一个（通常是奇异的）超曲面上、并投影到第一个坐标时的临界点；该超曲面由 $\mathbb{K}[x_1, \ldots, x_{n+1}]$ 中两个随机稠密二次型A、B 的结式定义：

$$f, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}, \quad f = \mathrm{resultant}(A, B, x_{n+1}).$$

7. 施泰纳多项式系统，源自 Breiding 等人（2020年）。

8. 其余所有示例均来自 BPAS 库（Asadi 等人，2021）。BPAS 库为 Maple 中的 RegularChains 库提供了一种替代方案，特别注重并行性。未来将其与我们的算法进行比较会很有意义。

为获得下表中的运行时间，我们几乎完全使用了基于见证集的数据结构来表示仿射胞腔。除 Bertini 外，每个多项式系统均在特征为 65521 的域上计算。Bertini 是一款数值软件，在复数域上进行计算。由于存在这一差异，比较 Bertini 与我们的运行时间时需要谨慎。我们在下表中将 Bertini 列标为灰色，以示区别。除 Magma 外，所有计算均在 Intel Xeon Gold 6244 CPU @ 3.60GHz 的单核上完成。所有 Magma 计算均在 Intel Xeon E5-2690 @ 2.90GHz 的单核上完成。我们让每种算法至少运行一小时，或运行完成相应方程组所需时间最快算法的 50 倍，取两者中较长者。

我们还利用输出中的见证集进行了以下工作，以便与 Bertini 比较：我们在一个较大的随机素特征下运行算法。随后，从每个输出分支中移除嵌入的不可约分支，并计算各输出分支的次数。由此可得输入所定义的代数集在各个维数上的次数。每当 Bertini 报告的次数不同时，我们便在相应列中作出标记。由于随机选取的特征较大，这些次数应当与在复数域上考察该代数集时得到的次数相同。

此外，在此表的第二列中，我们给出了算法等维 将相应方程组分解成的仿射胞腔数量。表中所有计时均以秒为单位。由于 Bertini 计时方式的限制，我们只能报告取整后的秒数，不保留小数位。

3.3 实验结果讨论

这里我们进一步介绍部分示例的相关信息，以及下文比较的不同实现方案在这些示例中的表现。

与其他实现相比，当输入方程组是稠密的（即方程组中的每个输入方程都涉及大多数或全部变量）时，我们的算法，也就是 等维算法，似乎表现最佳。

问题中的情形包括 cyclic 8、Ps(●) 方程组类、Sing(●) 方程组类、sos(●, ●) 方程组类以及 Steiner 多项式系统。

对于某些多项式系统，其中每个输入方程只涉及少数几个变量，我们通过不采用基于见证集的数据结构，转而运行类似于第 2 节中版本的确定性算法，成功缩短了运行时间。之所以能取得这样的改进，是因为用随机超平面与非常稀疏的系统相交可能会“破坏其稀疏性”，使某些格罗布纳基计算变得困难得多。Leykin-1 就是一个这样的例子：运行确定性版本后，耗时缩短至 2.6 秒。

Gonnet 和 dgp6 多项式系统表明，我们的算法对输入方程的排列顺序非常敏感：默认情况下，我们在算法 等维 中按次数逐次遍历输入方程。按照这种顺序，我们的算法经过数小时计算仍未能终止。而在这两个例子中调整顺序，改为按支撑长度对输入方程排序后，算法均在不到一秒内终止。我们算法的实际效率很大程度上取决于其遇到的中间多项式系统的难度，而这些系统的难度又取决于输入方程的顺序。在 Gonnet 多项式系统中，新顺序使最初几个中间系统仅涉及部分变量，因此更易于计算其格罗布纳基。在 dgp6 例子中，新顺序产生的一个中间多项式系统由单项式和二项式构成；我们的算法会对其进行非常精细的分解，从而显著简化其余方程的处理

对于方程组 sys2874，可以通过调整输入方程的顺序，使其按支撑长度排序；也可以使用我们算法的确定性版本来处理。这样一来，运行时间提升了数个数量级，降至 0.26 秒。

我们还注意到，对于 sys2449、sys2297 和 Leykin-1 这些例子，如果分解的是这些方程组的根基，而不是方程组本身，Oscar 的运行时间会显著缩短（均低于一秒）。

对于 KdV 和 sys2882 这两个例子，瓶颈似乎在于极其困难的格罗布纳基计算，而非我们算法的内在结构。我们曾进行非正式实验，尝试用 msolve 仅计算这些方程组的格罗布纳基；结果表明，即便如此，这也是一项极具挑战性的计算。对于这两个方程组，涉及正则链的技术似乎远胜于任何涉及格罗布纳基计算的方法。

总而言之，这些实验表明，在各种各样的例子中，我们的算法平均而言优于最先进的实现，并且能够解决一些此前无法解决的问题

致谢

作者谨向 Marc Moreno Maza 致谢，感谢他对 BPAS 库及其基准测试示例提出反馈。此外，作者也感谢两位匿名审稿人提出的宝贵意见，并认真审阅了本稿。

本工作得到法国国家研究署（ANR）的资助，资助协议编号为 ANR-18-CE33-0011（SESAME）、ANR-19-CE40-0018（Re Rerum Natura）；得到 ANR 与奥地利科学基金 FWF 的联合资助，资助协议编号为 ANR-19-CE48-0015（ECARP）和 ANR-22-CE91-0007（EAGLES）；得到 EOARD-AFOSR 资助，资助协议编号为 FA8665-20-1-7029；得到德国研究联合会特别研究领域 TRR 195、莱茵兰-普法尔茨州研究倡议的资助；并得到欧洲研究理事会（ERC）在欧盟地平线欧洲研究与创新计划框架下的资助，资助协议编号为 101040794（10000 DIGITS）。

name	数值域计算	等维	Maple	Oscar	Magma	Magma（素数分解）	Bertini
8-3-config-Li	23	1.6	16 ×10	> 1h	> 1h	65 ×40	4 ×2.5
cyclic8	6	381	> 5h	> 5h	> 5h	> 5h	⚠ 126 ×0.3
dgp6	3	0.2	53	2.2	> 1h	1.2	75
Gonnet	3	0.2	2.1	2.8	> 1h	1.4	74
P4L1	6	0.3	2.4	1.8	0.7	1.5	⚠ 21
P4L3	8	0.3	3.3	10	< 0.1	1.5	11
KdV		> 4h	353	> 4h	> 4h	7109 ×20	> 4h
Leykin-1	13	2.6 ×1.9	4 ×3.2	641 ×468	> 1h	1.4	✖
C1	4	129	> 1h	> 1h	> 1h	> 1h	✖
C2	4	0.3	100	152	> 1h	2.0	✖
C3	13	10	55	7	0.3	1.5	✖
MontesS16	6	1.9 ×1.4	2.7 ×1.9	2.0 ×1.4	1.4	1.5 ×1.1	7 ×5
Ps(10)	2	1.7	> 1h	30 ×17	> 1h	6 ×3.3	9 ×5
Ps(12)	2	51	> 1h	> 1h	> 1h	2060 ×40	⚠ 38 ×0.7
Ps(6)	2	< 0.1	0.2	1.7	0.5	0.3	2.0
Ps(8)	2	< 0.1	4	1.7	1.2	0.8	6
Sing(10)	2	0.4	> 1h	> 1h	> 1h	> 1h	⚠ 495
Sing(4)	2	< 0.1	76	2.2	8	5	5
Sing(5)	2	< 0.1	> 1h	4	7	1636	⚠ 1.0
Sing(6)	2	< 0.1	> 1h	51	> 1h	> 1h	⚠ 8
Sing(7)	2	< 0.1	1704	399	> 1h	> 1h	54
Sing(8)	2	0.1	> 1h	995	> 1h	> 1h	139
Sing(9)	2	0.2	> 1h	> 1h	> 1h	> 1h	⚠ 271
sos(4,2)	2	< 0.1	16	2.2	1.3	1.2	1.0
sos(4,3)	2	< 0.1	694	2.6	3.4	6	3.0
sos(5,2)	2	< 0.1	> 1h	1.8	3.7	1.2	3.0
sos(5,3)	2	< 0.1	> 1h	> 1h	> 1h	149	15
sos(5,4)	2	0.5	> 1h	> 1h	> 1h	> 1h	21
sos(6,2)	2	< 0.1	> 1h	2.0	5	1.6	5
sos(6,3)	2	0.1	> 1h	> 1h	> 1h	> 1h	34
sos(6,4)	2	5	> 1h	> 1h	> 1h	> 1h	69 ×14
sos(6,5)	2	14	> 1h	> 1h	> 1h	> 1h	⚠ 40 ×2.9
施泰纳	2	870	> 12h	> 12h	> 12h	> 12h	✖
sys2128	20	1.1	9 ×9	5 ×4	> 1h	1.9 ×1.8	829 ×790
sys2161	33	8	29 ×3.8	> 1h	> 1h	8 ×1.0	1196 ×159
sys2297	11	0.5	14	49	> 1h	1.7	⚠ 497
sys2353	13	1.6 ×1.2	5 ×3.8	2.0 ×1.5	5 ×3.7	1.3	> 1h
sys2449	24	1.3	28 ×21	60 ×46	> 1h	2.2 ×1.7	1338 ×1014
sys26 47	2	< 0.1	7	4	> 1h	2.0	⚠ 9
sys2874	5	0.3	202	1.9	8	10	> 1h
sys2880	50	4 ×2.4	144 ×80	1.8	3.4 ×1.9	4 ×2.2	⚠ 324 ×180
sys2882		> 1h	39	> 1h	> 1h	> 1h	✖
sys2885	2	0.3	6	2.2	> 1h	1.2	76
sys2945	5	0.5	3.2	1.8	0.6	1.0	120
sys2946	7	0.2	0.7	2.1	> 1h	1.6	3.0
W2	4	0.9	6	1.9	7	1.0	61
W44	3	0.5	13	4	> 1h	1.5	⚠ 66
Wu-Wang	3	3.1	3.2	1.8	0.7	1.3	106

除非另有说明，时间均以秒为单位。当最佳时间超过 1 秒时，给出相对于最佳时间的比值。

« 我们对输入进行了少量预处理（例如重新排列输入方程，或禁用仿射胞腔的概率表示），以改善运行时间。
é Bertini 因错误而终止了计算。 o Bertini 给出的结果在次数 / 维数方面与我们的结果不一致。

参考文献

Asadi, M., Brandt, A., Chen, C., Covanov, S., Kazemi, M., Mansouri, F., Mohajerani, D., Moir, R. H. C., Moreno Maza, M., Talaashrafi, D., Wang, L., Xie, N., & Xie, Y.（2021年）。Basic Polynomial Algebra Sub-programs (BPAS)（版本 1.791）。Aubry, P., Lazard, D., & Moreno Maza, M.（1999年）。关于三角集理论。J. Symb. Comput., 28(1), 105–124。Aubry, P., Rouillier, F., & Safey El Din, M.（2002）。正维系统的实数求解。J. Symb. Comput., 34(6), 543–560。Bardet, M., Faugère, J.-C., & Salvy, B.（2015年）。关于 F5 格罗布纳基算法的复杂性。J. Symb. Comput., 70, 49–70。Bates, D. J., Hauenstein, J. D., Sommese, A. J., & Wampler, C. W.（2013）。使用 Bertini 数值求解多项式系统。SIAM。Becker, T., & Weispfenning, V.（1993）。格罗布纳基（第 141 卷）。施普林格出版社。Berthomieu, J., Eder, C., & Safey El Din, M.（2021）。Msolve：用于求解多项式系统的库。ISSAC 2021 会议录。Berthomieu, J., Eder, C., & Safey El Din, M.（2022）。计算饱和理想的格罗布纳基的新型高效算法（F4SAT）以及商理想的格罗布纳基的新型高效算法（Sparse-FGLM-colon）。arXiv: 2202.13387。Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B.（2017）。Julia：数值计算的新方法。SIAM Rev., 59(1), 65–98。Bosma, W., Cannon, J., & Playoust, C.（1997）。Magma 代数系统。I. 用户语言。J. Symb. Comput., 24(3-4), 235–265。Breiding, P., Sturmfels, B., & Timme, S.（2020）。一秒钟内得到 3264 条圆锥曲线。Not. Am. Math. Soc., 67(1), 30–37。Caboara, M., Conti, P., & Traverse, C.（1997）。又一种理想分解算法。AAECC 1997, 39–54。Chen, C., Corless, R. M., Moreno Maza, M., Yu, P., & Zhang, Y.（2013）。正则链理论在极限环研究中的应用。Int. J. Bifurc. Chaos, 23(09), 1350154。Chen, C., Lemaire, F., Moreno Maza, M., Pan, W., & Xie, Y.（2007）。使用 RegularChains 库高效计算无冗余三角分解。Int. Conf. Comput. Sci., 268–271。Chen, C., & Moreno Maza, M.（2012）。计算多项式系统三角分解的算法。J. Symb. Comput., 47(6), 610–642。Chou, S.-C., & Gao, X.-S.（1990）。Ritt-Wu 分解算法与几何定理证明。第 10 届自动演绎国际会议，207–220。Cox, D. A., Little, J., & O’Shea, D.（2015年）。理想、代数簇与算法（第 4 版）。施普林格出版社。Decker, W., Greuel, G.-M., & Pfister, G.（1999年）。主分解：算法与比较。算法代数数论，187–220。Decker, W., Greuel, G.-M., Pfister, G., & Schönemann, H.（2021）。Singular 4-2-1——用于多项式计算的计算机代数系统。Eder, C., Lairez, P., Mohr, R., & Safey El Din, M.（2022）。用于计算多项式系统非退化轨迹的基于签名的算法。arXiv: 2202.13784。Eisenbud, D., Huneke, C., & Vasconcelos, W.（1992）。主分解的直接方法。Invent. Math., 110(1), 207–235。Gallo, G., & Mishra, B.（1991年）。Wu-Ritt 特征集的高效算法与上界。载于 T. Mora & C. Traverso（编），代数几何中的有效方法（第 119–142 页）。Birkhäuser。García Fontán, J., Nayak, A., Briot, S., & Safey El Din, M.（2022）。透视四线与五线问题的奇异性分析。国际计算机视觉杂志，130, 909–932。Gianni, P., Trager, B., & Zacharias, G.（1988）。多项式理想的格罗布纳基与主分解。J. Symb. Comput., 6(2), 149–167。Hashemi, A., & Seiler, W. M.（2017）。依赖维数的格罗布纳基上界。ISSAC 2017 会议录，189–196。Hubert, E.（2003）。三角集与三角化分解算法笔记 I。Symb. Numer. Sci. Comput., 1–39。Ishihara, Y.（2022）。中间主分解的模技术。ISSAC 2022 会议录，479–487。

Jeronimo, G., & Sabia, J.（2002）。仿射簇的有效等维分解。纯粹与应用代数杂志，*169*(2), 229–248。Kalkbrener, M.（1993）。计算代数簇三角表示的广义欧几里得算法。符号计算杂志，*15*(2), 143–167。Kalkbrener, M.（1994）。多项式环中根基的素分解。符号计算杂志，*18*(4), 365–372。Krick, T., & Logar, A.（1991年）。环中理想根基的计算算法。载于 *AAECC 1991*年 （第 195–205 页）。施普林格出版社。Lazard, D.（1991年）。求解正维代数系统的新方法。离散应用数学，*33*(1), 147–160。Lecerf, G.（2000）。借助几何分解计算代数簇的等维分解。*ISSAC* 会议录 *2000*，209–216。Lecerf, G.（2003）。借助提升纤维计算代数闭集的等维分解。复杂性杂志，*19*(4), 564–596。Lemaire, F., Moreno Maza, M., & Xie, Y.（ 2005 ）。MAPLE 中的 RegularChains 库。*ACM SIGSAM* 通讯，*39*(3), 96–97。Lu, Y., & Jingzhong, Z.（1994）。搜索代数方程之间的依赖关系：一种应用于自动推理的算法。Moroz, G.（2008）。正则分解。载于 D. Kapur（编），计算数学 （第 263–277 页）。施普林格出版社。Noro, M., & Yokoyama, K.（2004）。小有限域上多项式理想素分解的实现。符号计算杂志，*38*(4), 1227–1246。Pascual-Escudero, B., Nayak, A., Briot, S., Kermorgant, O., Martinet, P., Safey El Din, M., & Chaumette, F.（2021年）。透视四点问题的完整奇异性分析。国际计算机视觉杂志，*129*(4), 1217–1237。Ritt, J. F.（1950）。微分代数。美国数学学会。Safey El Din, M., & Schost, É.（2004）。投影的适当性缺陷以及在实代数集中每个连通分支中至少计算一个点。离散与计算几何，*32*(3), 417–430。Safey El Din, M., Yang, Z.-H., & Zhi, L.（2018）。计算多项式系统实根基的复杂性。*ISSAC* 会议录 *2018*，351–358。Safey El Din, M., Yang, Z.-H., & Zhi, L.（2021）。计算多项式系统的实根基与 S-根基。符号计算杂志，*102*，259–278。Sommese, A. J., Verschelde, J., & Wampler, C. W.（2005）。数值代数几何导论。载于 M. Bronstein, A. M. Cohen, H. Cohen, D. Eisenbud, B. Sturmfels, A. Dickenstein, & I. Z. Emiris（编），求解多项式方程 （第 301–337 页）。施普林格出版社。OSCAR 团队。（2023）。*OSCAR*——开源计算机代数研究系统。Wang, D.（1993）。多项式系统的消元方法。符号计算杂志，*16*(2), 83–114。Wang, D.（2001）。消元方法。施普林格出版社。Wu, W., & Gao, X.（2007）。三十年后的数学机械化及其应用。中国计算机科学前沿，*1*(1), 1–8。Wu, W.-T.（1986）。初等几何机械化定理证明的基本原理。*2*(3), 221–252。Yang, L., Hou, X., & Xia, B.（1998）。几何不等式的自动发现与证明。自动演绎几何国际研讨会，30–46。Yang, L., Hou, X., & Xia, B.（2001）。一类不等式型定理自动发现的完整算法。中国科学：信息科学，*44*(1), 33–49。Yokoyama, K.（2002）。有限域上多项式理想的素分解。载于数学软件（第 217–227 页）。世界科技出版公司。